Lingua-Align: An Experimental Toolbox for Automatic Tree-to-Tree Alignment

Jörg Tiedemann

Department of Linguistics and Philology
Uppsala University, Uppsala/Sweden
jorg.tiedemann@lingfil.uu.se

Abstract

In this paper we present an experimental toolbox for automatic tree-to-tree alignment based on local classification and alignment inference. The aligner implements a recurrent architecture for structural prediction using history features and a sequential classification procedure. The discriminative base classifier uses a log-linear model which enables simple integration of various features extracted from the data. The Lingua-Align toolbox provides a flexible framework for feature extraction including contextual properties and implements several alignment inference procedures. Various settings and constraints can be controlled via a simple frontend or called from external scripts. Lingua-Align supports different treebank formats and includes additional tools for conversion and evaluation. In our experiments we can show that our tree aligner produces results with high quality and outperforms unsupervised techniques proposed otherwise. It also integrates well with another existing tool for manual tree alignment which makes it possible to quickly integrate additional training material and to run semi-automatic alignment strategies.

1. Introduction

Most data-driven machine translation (MT) approaches are based on knowledge extracted from parallel corpora. In recent years one could see a shift towards the integration of syntactic information into such systems. This includes various syntax-enhanced approaches to statistical MT and also applies to example-based MT in which translation templates can be extracted from aligned syntactic structures.

There is a distinction between techniques that exploit linguistically motivated structures and the ones that induce syntactic mappings (for example synchronous grammars) from raw text specifically optimized for the task of translation. In the grammar induction approach tree alignment is the result of training and is often based on existing word alignments (Chiang, 2007) or iterative re-estimation procedures (Wu, 1995; Saers et al., 2009). Linguistically motivated structures can be integrated in various ways, for example, by defining syntactic constraints in training (Zollmann and Venugopal, 2006; Chiang et al., 2008; Chiang et al., 2009). Other approaches rely entirely on the mapping between linguistic parse trees that can be extracted from parallel aligned treebanks (Poutsma, 2000; Vandeghinste and Martens, 2009). However, only a few resources exist and most of them are created with a large human effort (Gustafson-Čapková et al., 2007; Ahrenberg, 2007; Gonzales et al., 2009). Therefore, techniques are required for the automatic alignment of parsed parallel corpora in order to produce appropriate amounts of training data.

In recent years some approaches to automatic tree alignment have been proposed (see, e.g., (Zhechev and Way, 2008; Lavie et al., 2008)) mostly using automatic word alignment to induce links between non-terminal nodes. In this paper we describe a discriminative approach based on standard classification techniques and supervised learning for this task. Discriminative models in general have the advantage that various feature can easily be integrated without considering the dependencies between them in the underlying data structure. Usually a loss function is iteratively optimized over a set of labeled training data in order to estimate model parameters. For tree alignment we observed that small amounts of training data seem to be sufficient in order to obtain reasonable performance. This encouraged us to further develop our toolbox which is described below. In this paper we include a description of the main concepts of our approach and some details of the implementation and usage of the software which includes tools for handling treebank data in various formats and scripts for evaluation.

2. Discriminative Tree Alignment

This section summarizes the general approach implemented in Lingua-Align. This includes a discussion of alignment features and link search strategies.

2.1. The General Approach

Tree alignment is modeled as a structured classification task using a discriminative feature-based model. In our approach we decompose the structured prediction problem into local binary decisions based on local, contextual and history features. In particular, we build a standard log-linear model for binary classification predicting links a_{ij} between node pairs s_i and t_j from the source and the target language tree, respectively:

$$P(a_{ij}|s_i,t_j) = \frac{1}{Z(s_i,t_j)} exp\left(\sum_k \lambda_k f_k(s_i,t_j,a_{ij})\right)$$

Here, $f_k(s_i,t_j,a_{ij})$ represent feature functions extracted from the candidate nodes and their contexts and the corresponding weights λ_k are learned in the training procedure. History features are embedded in the procedure using a so-called "recurrent architecture" (Dietterich, 2002) in which prediction is performed sequentially incorporating previous decisions as additional features. Classification is performed in a bottom-up manner using predictions on child nodes as our history. In particular, we count the number of linked nodes dominated by the current node pair and

normalize this count with the number of child nodes of the larger subtree. In classification we use link likelihoods as "soft counts" instead of local link decisions made by the classifier in order to incorporate uncertainties of the classifier. In training we may also include a variant of SEARN (Daumé III, 2006) in order to reduce the label bias problem (van den Bosch, 1997) which is a known issue in recurrent prediction strategies (see also section 2.3.). This iterative strategy can be switched on or off using simple runtime parameters. In a final step, we use the likelihoods of links predicted by the classifier in an additional inference procedure that performs the actual alignment (see section 2.4.).
Using this general approach we are able to align any pair of tree structures without being bound to specific types of syntactic annotation and grammar formalisms. Furthermore, we may also switch off the history features and use a simple greedy alignment strategy without bottom-up classification and well-formedness checks which would allow us to align even non-tree structures (for example dependency graphs) without changing the general approach. This flexibility makes our tree aligner a valuable resource for various kinds of applications in which structural alignment is required. However, most of its power comes from the feature extraction which is optimized for certain types of data structures and the ability of the local classifier to learn appropriate predictions. In the next section we briefly describe the features used in our current implementation.

2.2. Alignment Features

Any real-valued feature function can be integrated in the log-linear model we are using. Here, we give a brief overview of features that we currently support in LinguaAlign. For illustration purposes we will use the example in figure 1 taken from SMULTRON, a multilingual aligned treebank (Gustafson-Čapková et al., 2007), which we will also use in our experiments described in section 4..

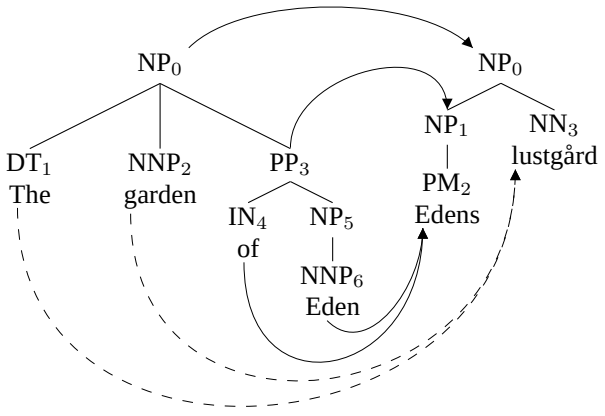

Figure 1: Example tree alignment from SMULTRON (Sophie's World, English-Swedish). The dashed lines refer to complementary links which are not part of the official treebank.

The first type of alignment features is similar to the lexical link scores introduced by (Zhechev and Way, 2008). This score combines lexical translation probabilities for given node pairs to form so-called *inside* scores (relations between leaf nodes within the subtrees) and *outside* scores (relations between leaf nodes outside of the current subtrees). We use a similar definition as (Zhechev and Way, 2008) with a slight modification, replacing mean link scores with the maximum link score:

$$\alpha_{inside}(s|t) = \prod_{s_i \in yield(s)} max_{t_j \in yield(t)} P(s_i|t_j)$$

$$\alpha_{outside}(s|t) = \prod_{s_i \notin yield(s)} max_{t_j \notin yield(t)} P(s_i|t_j)$$

The lexical translation probabilities $P(s_i|t_j)$ used above are usually extracted from automatic word alignment. The inverse translation table with $P(t_j|s_i)$ probability estimations can also be applied to compute inverse inside/outside scores, accordingly. In contrast to (Zhechev and Way, 2008), we use these four individual scores as separate feature functions which makes it more flexible for the learning algorithm to adjust the importance of these values.
A second type of features uses existing word alignments directly by computing link consistency scores for given tree node pairs. This score is simply the proportion of word alignments which are consistent with the chosen subtree pair among all the other ones involving words covered by these subtrees. Any kind of word alignment can be used to compute these scores. We usually apply the directional Viterbi alignments of IBM model 4 (produced by Giza++ (Och and Ney, 2003)) and symmetrized alignments produced by Moses (Koehn et al., 2007) on top of the IBM models. For example, consider the following (mostly incorrect) word alignment produced by Giza++ for the sentence pair from our example:

```
the garden of eden
NULL {} edens {1} lustgård {2 3 4}
edens lustgård
NULL {} the {} garden {} of {} eden {1 2}
```

This would create the following features for some of the nodes in our example tree pair:

source	target	$GIZA_{src2trg}$	$GIZA_{trg2src}$
NNP_6	PM_2	0	0.5
NNP_6	NN_3	1/3	0.5
NP_0	NP_0	1	1
NP_0	NP_1	0.25	0.5

Table 1: Examples of word alignment features

A third type of features considers the positions of the nodes within the parse trees. Firstly, we define a *tree-level similarity* score which compares the relative vertical positions of the current nodes within their parse trees. For this we normalize the distance to the root node with the maximum distance of any leaf node from the root in that tree and compute the absolute difference between these relative positions. Secondly, we define a *tree-span similarity* score by comparing the relative horizontal positions of the subtrees. For this we compute the mean of the positions in the sentence of all words dominated by the current node and

calculate relative positions by normalizing this value with the overall length of the sentence. Finally, we also compute the *ratio of leafs* dominated by the two tree nodes as another tree structure feature. Table 2 shows a number of such features extracted from the example tree pair above.

source	target	span-sim	level-sim	leaf-ratio
NNP_6	PM_2	0	1	1
NNP_6	NN_3	1	0.5	1
NP_0	NP_0	1	1	0.5
NP_0	NP_1	0.5	0.5	0.25

Table 2: Examples of tree structure features

The last type of base features considers labels in the parse trees – phrase structure categories for non-terminal nodes and part-of-speech labels for terminal nodes. These features are typically very important for the classification model but language-pair specific. Each of them is used as a binary feature and appropriate weights are learned from the training data. Other, similar types of labels can also easily be integrated, for example, dependency labels.

It is important to mention that all the features described above can also be extracted from any contextual node. We implemented a simple formalism to select features from arbitrary nodes connected with the current node pair. This is done by adding a prefix to the feature template specifying the movement that has to be performed before extracting the corresponding feature value. For example, a prefix "parent" refers to moving to the immediate parent nodes of both, the source and the target language node. The prefix "srcparent" refers to a movement on the source language side only (similarly with "trgparent" for the target language tree). With the prefix "children" we may move to the daughter nodes and retrieve the mean of the according feature values. The prefix "sister" makes it possible to move to nodes with the same parent. Again there are variants of these operations for moving on one side only. All these operations can be applied recursively, such that we can move, for example, to the "grandparent" node by specifying a prefix "parent_parent". Corresponding features are only extracted if all movements can be carried out. Table 3 shows some possible contextual features of a particular pair of nodes extracted from our example trees.

feature	value
sister_labels=PP-NP	1
sister_labels=NNP-NP	1
parent_$\alpha_{inside}(t\|s)$	0.00001077
srcparent_$GIZA_{src2trg}$	0.75

Table 3: Contextual features for node pair $\langle DT_1, NN_3 \rangle$

The use of contextual features is one way to implicitly integrate structural dependencies in the alignment process. Note that contextual features may indicate negative cues even in cases where the same type of feature would represent a positive one for the current node pair. For instance, a strong "srcsister_$GIZA_{src2trg}$" feature may indicate that a link between a sister node on the source language side and the current target language node should be preferred over a link between the current node pair whereas the "$GIZA_{src2trg}$" feature usually represents a positive cue for the current node pair. It is up to the learning algorithm to adjust the parameters accordingly.

Finally, it is also allowed to combine base features like the ones above to create complex features in order to account for any non-linear correlation between them within the classification task. Examples of complex features are combinations of category labels and word alignment scores or tree-span similarity scores combined with tree-level similarity scores. In case of real-valued features we simply multiply corresponding values. Other types of score combinations could be integrated as well but are not supported at the moment.

Lingua-Align provides in this way a flexible framework for feature extraction for the tree alignment task supporting a set of base features that can be retrieved from any contextual node which may be combined with each other in various ways. Using this framework different experiments can easily be carried out and the alignment process can be adjusted according to particular settings and needs (for example, adjusting alignment speed versus alignment quality or using the software with or without word alignment information).

2.3. Training the Local Classifier

Any standard classifier that supports real-valued features can be used for our task once appropriate features have been extracted from the data. In fact it would not require more than minor adjustments in the feature representation to call a different training and classification algorithm implemented in any kind of external machine learning package or library. In our current setup we opted for a log-linear model and a maximum entropy classifier implemented in the freely available toolbox Megam (Daumé III, 2004). This toolbox includes very efficient training procedure that allow fast experimentation with large feature sets. In fact the bottleneck in training (and testing) is not the local classifier but the extraction of features from the data which often requires quite a lot of tree traversal steps and other operations on the underlying data structure.

Basically we use standard settings for training the binary classifier using real-valued features and the conjugate gradient algorithm implemented in Megam. Other settings can be adjusted by adding appropriate arguments to the call of the external software. Future releases of Lingua-Align will probably support other binary classifiers and external machine learning packages as well to make it possible to test various kinds of learning approaches.

It is important to remember that the approach of using local classification for tree alignment is only an approximation of the underlying complex structured prediction problem. There are some possible issues with the decomposition of the global problem into sequential decisions. History features are valuable cues for structural dependencies. However, the risk of error propagation should not be neglected when depending on local predictions. In our case, we therefore do not include hard decisions of the classifier to set the

values of history features. Instead we use the prediction likelihood for each possible link candidate as a soft count for subsequent predictions. In this way, uncertainties of the classifier influence future decisions. However, there is also the problem of learning with history features which may lead to the label bias problem (van den Bosch, 1997). The fact that we learn from correctly annotated training data produces decision problems when confronting the classifier with erroneous and uncertain features in the actual annotation step. Therefore, it may be a good idea to adjust the local classifier to the "real situation" in practical prediction. This can be done using a simple iterative training procedure. We adapted ideas of SEARN ("search-learn") (Daumé III, 2006) in which history features are interpolated with the predictions of the current classifier to train a new classifier for the next iteration. In SEARN the optimal interpolation weight should be learned on some development data. This step is skipped in our current implementation which may explain the unsatisfactory results in the experiments described below in section 4.. This will be investigated further in future work.

2.4. Alignment Inference

After training a classifier this model can be applied for predicting actual links between given node pairs. As mentioned earlier, we do not need to rely directly on the local predictions but may include an additional alignment inference procedure in order to integrate additional structural constraints.

2.4.1. Maximum weight matching

One common constraint is a restriction to one-to-one alignments which seems to be reasonable for linking non-terminal nodes. Using prediction likelihoods as link scores we can apply standard search strategies to infer links between the nodes of the entire tree structure. One possibility is to use well-known graph-theoretic algorithms modeling alignment as an assignment problem in a weighted bipartite graph. In graph theory a bipartite graph is characterized by the property that its nodes can be divided into two disjoint sets S and T such that every edge in the graph connects a node in S to one in T. In our case S refers to the nodes in the source language tree and T refers to the nodes in the target language tree. Edges in the bipartite graph refer to alignments between S and T. Using our local classifier we can assign scores $p_{ij} = P(a_{ij}|s_i, t_j)$ for every possible link a_{ij} creating a weighted bipartite graph. The assignment problem is now to find a maximum weight matching in this graph, i.e. the set of edges without common nodes that maximize the sum of the values attached to them. This task can also be interpreted as the assignment $\vec{a} = (a_1 \cdots a_n)^T$ of n "tasks" (source tree nodes) to n "agents" (target tree nodes) that minimizes the overall cost $C = \sum_{i=1}^{n} c_{ia_i}$ where each possible assignment has the cost $c_{ij} = 1 - p_{ij}$:

$$assign\left(\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}\right) = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

This fundamental combinatorial optimization problem can be solved by a number of well-known algorithms. The Hungarian method (Kuhn, 1955) is one of them solving this task in polynomial time (Munkres, 1957). Several implementations of this algorithm (also know as the Kuhn-Munkres algorithm) exist and in our case we integrated a publicly available Perl extension (Kulkarni and Pedersen, 2008). This module extends the original algorithm to handle cases in which the number of nodes differs in both sets by padding zeros in the corresponding matrix. This basically leads to some unaligned nodes in either source or target language tree which is, of course, necessary in the case of a one-to-one link constraint.

2.4.2. Greedy alignment

Another way of handling one-to-one alignment constraints is to run a simple greedy alignment strategy as proposed by (Melamed, 2001) for word alignment. In this case, we simply start with the highest score among all link candidates and align the corresponding nodes with each other. We then remove these nodes from the search space and continue with the next ones until no more links can be added. In this way we also obtain strictly one-to-one alignments with possibly some unaligned nodes on one side. However, we are not guaranteed to obtain the maximum weight matching as in the assignment algorithm explained above. On the other hand, an advantage of the greedy strategy is that other constraints can easily be added to this competitive linking approach. For example, additional well-formedness constraints can be included as defined by (Zhechev and Way, 2008). These constraints basically add the following restriction: Descendants/ancestors of a source linked node may only be linked to descendants/ancestors of its target linked counterpart. In our experimental results we will see the importance of these additional constraints when applied to tree alignment. This leads to a clear advantage of the greedy strategy compared to the assignment algorithm from the previous section which cannot guarantee this kind of well-formedness. Figure 2 summarizes the basic greedy search algorithm.

```
1: sorted ←sort_indeces_by_value(P(a_ij|s_i, t_j))
2: while (i, j) ← pop sorted do
3:    if P(a_ij|s_i, t_j) < min_score then
4:       return links
5:    end if
6:    if not srcLink[i] and not trgLink[j] then
7:       if is_wellformed(i, j, links) then
8:          links[i][j] ← P(a_ij|s_i, t_j)
9:          srcLink[i] ← true
10:         trgLink[j] ← true
11:      end if
12:   end if
13: end while
14: return links
```

Figure 2: Greedy search with well-formedness constraints

The greedy algorithm can be adjusted in various ways. For example, it is possible to relax the well-formedness con-

straints to allow multiple links per node by replacing line 6 in the algorithm above with "**if not** $srcLink[i]$ **or not** $trgLink[j]$ **then**" and by adjusting the *is_wellformed* function accordingly. Other constraints can also easily be integrated which makes this strategy a very flexible approach for alignment inference.

2.4.3. Combined strategies

There are many other ways for inference with alignment heuristics like the greedy competitive approach. For example, one can define source-to-target matching algorithms with additional well-formedness constraints. Similarly this can be applied in the opposite direction. Furthermore, directional alignments can be combined again using heuristics similar to the ones applied in traditional word alignment using asymmetric IBM models (symmetrization heuristics such as intersection etc). Another strategy is to split alignment into several steps, for example aligning non-terminal nodes first and terminal nodes thereafter using well-formedness constraints in both steps. It is also possible to use a restrictive alignment strategy first (for example the assignment algorithm with subsequent removal of non-wellformed links) followed by a relaxed strategy with less constraints. Furthermore, additional constraints can be added, for example, forcing the aligner to link nodes of the same type only (non-terminals with non-terminals and terminal nodes with terminal nodes). One may also skip the alignment of unary tree productions.

Lingua-Align implements a number of strategies and allows various combinations of constraints and cascaded alignment steps. In this way, the alignment result can easily be adjusted according to one's needs (focusing on precision or recall, requiring complete alignment on one side or both sides, etc). Table 4 illustrates the impact of inference strategies by comparing links proposed by different base strategies for the example tree pair from figure 1 (none of the final alignments is really satisfactory).

source	target	threshold	greedy	munkres	wellformed
NP_0	NP_0	yes	yes	yes	yes
PP_3	NP_1	-	yes	yes	-
NNP_6	NN_3	-	yes	yes	yes
NNP_2	PM_2	-	yes	yes	yes

Table 4: Alignments proposed by different inference strategies (*threshold* refers to the decisions made by the local classifier; *munkres* is the combinatorial assignment result; *wellformed* is a greedy search strategy with well-formedness constraints)

3. The Tree Aligner Package

Lingua-Align is a collection of Perl modules and a number of simple frontend scripts that can be used to call the training and alignment procedures. The tree aligner is intended to be used on the command line supporting a variety of parameters to adjust the behavior of the alignment process.

The modular design makes it also straightforward to integrate Lingua-Align in other software and to extend it with additional modules. The library structure is divided into modules that handle the interaction with the external classifier (training and classification), modules that manage various kinds of data formats (treebanks, parallel corpora, word alignments), modules for feature extraction and modules that perform different kinds of alignment inference algorithms. The implementation uses an object-oriented style to make it easy to extend the package.

The main frontend can be used for both training a new model and aligning a given parallel treebank. Command line arguments can be specified to set appropriate parameters for the training procedure and the alignment phase. The classification model is stored in a plain text file and can be re-used for any subsequent alignments. Another advantage of the discriminative approach is that the alignment of large treebanks can easily be run in parallel once the model has been trained. The main bottleneck is feature extraction but this can be done individually for each tree pair. In this way, many partitions of a treebank can be aligned simultaneously using the same alignment model.

```
<?xml version="1.0" ?>
<treealign>
<treebanks>
 <treebank id="en"
  filename="SMULTRON_EN_Sophies_World.xml"/>
 <treebank id="sv"
  filename="SMULTRON_SV_Sophies_World.xml"/>
</treebanks>
<alignments>
 <align author="Lingua::Align" type="fuzzy"
        prob="0.44271032770176010906">
  <node node_id="s1_4" treebank_id="en"/>
  <node node_id="s1_2" treebank_id="sv"/>
 </align>
 <align author="Lingua::Align" type="good"
        prob="0.75750083118538613647">
  <node node_id="s1_502" treebank_id="en"/>
  <node node_id="s1_500" treebank_id="sv"/>
 </align>
 <align author="Lingua::Align" type="fuzzy"
        prob="0.17420817390945234071">
  <node node_id="s1_2" treebank_id="en"/>
  <node node_id="s1_1" treebank_id="sv"/>
 </align>
 <align author="Lingua::Align" type="fuzzy"
        prob="0.28574840542833246371">
  <node node_id="s1_501" treebank_id="en"/>
  <node node_id="s1_501" treebank_id="sv"/>
 </align>
</alignments>
</treealign>
```

Figure 3: An example of the Tree Aligner Format

One difficulty is the dependence of some important features on prior word alignment. This basically requires to run statistical word alignment on the entire corpus in order to prepare feature extraction. Lingua-Align supports the common file formats produced by Giza++ and Moses (Viterbi alignment, lexical translation table, symmetrized word alignment) and reads these files natively when producing tree alignment features. Important here is to pro-

vide appropriate files that match the sentences in the parallel treebank to be aligned.

The standard output format of Lingua-Align is similar to the one used by the Stockholm Tree Aligner (STA), a tool for manual alignment (Lundborg et al., 2007). Figure 3 shows a short example of the alignment format produced by our tool. This format is also expected for the training data and the default format for corpus encoding is TigerXML as it is for STA.

Using the same file conventions as STA makes it straightforward to integrate manual and automatic tree alignment. This is especially useful for the rapid development of new training data and the manual inspection of automatic alignment results. Furthermore, it is straightforward to use the result of automatic tree alignment in STA for manual post-editing in order to produce validated parallel treebanks in a semi-automatic way. A closer integration of the automatic aligner into the graphical interface of STA could be an interesting direction for future work (possibly in connection with an on-line learning algorithm for incremental improvements of the basic classifier).

Lingua-Align also supports two other treebank formats: AlpinoXML and Penn treebank format. These formats can directly be used by the alignment software. Conversion tools are also included to change the annotation from one format to another.

Finally, there is also a script for alignment evaluation. We use standard measures of precision, recall and balanced F-scores defined as follows:

$$Precision = \frac{|P \cap A|}{|A|} \quad Recall = \frac{|S \cap A|}{|S|}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

These metrics are adapted from the word alignment literature (Och and Ney, 2003) using a distinction between "sure" (S) links and "possible" (P) links. In SMULTRON there is a distinction between "good" and "fuzzy" links. We interpret "good" links as "sure" and add "fuzzy" links to the set of "good" links to describe the set of "possible" links. A refers to the links proposed by the system (without making a distinction between good and fuzzy links). The evaluation script computes these basic scores for a given alignment and a given gold standard and also provides separate numbers for non-terminal and terminal nodes. An example output of the evaluation script can be seen in figure 4[1].

4. Experiments

We tested our package with data taken from the SMULTRON treebank. In particular, we used the two chapters of "Sophie's World" in Swedish and English to train and test various tree alignment models which includes 6,671 "good" links and 1,141 "fuzzy" links between nodes in the parse trees of about 500 sentence pairs. We used the first 100 sentences for training and tested alignment quality on the remaining part of the corpus. In this way we use a larger portion of the corpus for evaluation in order to obtain reliable scores. Alignment quality is measuered using precision, recall and F-score as described above. The results are shown in table 5.

```
----------------------------------------
   precision (ALL/NT:NT) = 82.32 (1984/2410)
      recall (ALL/NT:NT) = 78.08 (1984/2541)
  balanced F (ALL/NT:NT) = 80.15
----------------------------------------
     precision (ALL/T:T) = 78.00 (2716/3482)
        recall (ALL/T:T) = 71.79 (2716/3783)
    balanced F (ALL/T:T) = 74.77
----------------------------------------
 precision (fuzzy/NT:NT) = 17.91 (65/363)
    recall (fuzzy/NT:NT) = 11.82 (65/550)
balanced F (fuzzy/NT:NT) = 14.24
----------------------------------------
   precision (fuzzy/T:T) =  6.91 (51/738)
      recall (fuzzy/T:T) = 16.89 (51/302)
  balanced F (fuzzy/T:T) =  9.81
----------------------------------------
  precision (good/NT:NT) = 71.42 (1462/2047)
     recall (good/NT:NT) = 73.84 (1462/1980)
 balanced F (good/NT:NT) = 72.61
----------------------------------------
    precision (good/T:T) = 81.74 (2243/2744)
       recall (good/T:T) = 64.44 (2243/3481)
   balanced F (good/T:T) = 72.06
========================================
  precision (all) = 79.77 (4700/5892)
     recall (all) = 74.32 (4700/6324)
    recall (good) = 75.66 (4132/5461)
   recall (fuzzy) = 65.82 (568/863)
========================================
F (P_all & R_all)  = 76.95
F (P_all & R_good) = 77.66
========================================
```

Figure 4: Tree alignment evaluation

The tree aligner produces very promising results. The performance consistently improves when adding features to the model. The advantage of our feature-rich discriminative approach can be seen when comparing the results with the performance of an unsupervised tree aligner. Running the subtree aligner described in (Zhechev and Way, 2008) on the same data set yields a balanced F-score of 57.57% which is similar to the scores we obtain when using the same lexical features. This scores is very low especially because of the size of the training data used for estimating lexical translation probabilities. However, when using a larger model (adding the entire Europarl corpus (Koehn, 2005) to our data set) the performance of the unsupervised tree aligner improves only slightly to about 58.64% (mainly due to the obvious domain mismatch).

Interesting is also to see the effect of different inference algorithms. The first two columns in table 5 refer to local classification without any additional inference procedures.

[1]Note that the proposed link type ("good" for links with certainty ≥ 0.5 and "fuzzy" for other links) does not influence the overall scores whereas it does for the individual scores of "good" and "fuzzy" links.

inference algorithm →	threshold=0.5		greedy		munkres		greedy+wellformed		
history features →	no	yes	no	yes	no	yes	no	yes	searn
lexical	38.52	40.00	50.05	56.76	49.75	56.60	52.03	**57.11**	57.05
+ tree	50.27	51.84	54.55	57.81	54.41	57.01	57.54	**58.68**	55.47
+ alignment	60.41	60.63	60.92	60.87	61.31	60.83	62.09	**62.88**	56.84
+ labels	72.44	72.24	72.94	73.14	72.72	73.05	75.72	**75.79**	73.88
+ context	74.68	74.90	75.03	75.60	74.96	75.38	77.29	**77.66**	75.21

Table 5: Results (balanced F-scores) for different feature sets and inference algorithms (greedy competitive linking, Kuhn-Munkres assignment, greedy linking with additional well-formedness constraints and the same with SEARN learning).

We can see that the performance of the basic classifier is much below the other inference-based approaches when using a limited set of features only. However, with increasing amounts of features the base classifier catches up with the inference algorithms which demonstrates the power of the local feature-rich log-linear model even in this structured prediction task. Furthermore, our test data does not seem to suffer from the label bias problem. The performance actually drops in all settings when applying our simplified version of SEARN training. This is probably due to the non-optimized fixed interpolation weight applied in our setup. We will investigate this strategy further in future work in order to see if we can obtain additional performance gains. The advantage of an inference-based approach can be seen when applying the well-formedness constraints which seems to be very important even in the feature-rich settings. The best performance is consistently achieved with the greedy search strategy using these constraints.

5. Conclusions

Lingua-Align is a toolbox for automatic tree-to-tree alignment that uses a local discriminative classification approach for a sequential alignment procedure. The toolbox includes a flexible framework for feature extraction from parallel treebanks supporting the extraction of contextual information and the integration of external tools for word alignment. History features can be used in a recurrent classification architecture to approximate the complex structured prediction task. Several alignment inference algorithms are implemented that can be run on top of the predictions of the local classifier.

In our experiments we can show that small amounts of training data are sufficient to obtain a reasonable tree aligner model. We are currently using the toolbox to align large-scale parallel treebanks in an on-going project on syntax-based machine translation (Tiedemann and Kotzé, 2009).

Lingua-Align also includes several tools for handling different kinds of data formats and for the evaluation of tree aligment results. It also integrates well with an existing tool for manual tree alignment which makes it possible to visualize and manually correct automatic alignment results.

6. References

Lars Ahrenberg. 2007. LinES: An English-Swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA, 2007)*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Morristown, NJ, USA. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Morristown, NJ, USA. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August.

Hal Daumé III. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California, Los Angeles, CA, August.

T.G. Dietterich. 2002. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30. Springer Verlag.

Annette Rios Gonzales, Anne Göhring, and Martin Volk. 2009. A Quechua-Spanish parallel treebank. In *Proceedings of TLT7*.

Sofia Gustafson-Čapková, Yvonne Samuelsson, and Martin Volk. 2007. SMULTRON (version 1.0) - The Stockholm MULtilingual parallel TReebank. http://www.ling.su.se/dali/research/smultron/index.htm. An English-German-Swedish parallel Treebank with sub-sentential alignments.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for

statistical machine translation. In *Proceedings of ACL*, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Harold W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Anagha Kulkarni and Ted Pedersen. 2008. Algorithm::Munkres - Perl extension for Munkres' solution to classical assignment problem for square and rectangular matrices. http://search.cpan.org/ tpederse/Algorithm-Munkres-0.08/lib/Algorithm/Munkres.pm.

Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation (SSST-2)*, pages 87–95, Columbus, Ohio, June. Association for Computational Linguistics.

Joakim Lundborg, Torsten Marek, Maël Mettler, and Martin Volk. 2007. Using the Stockholm TreeAligner. In *Proceedings of the 6th Workshop on Treebanks and Linguistic Theories*, pages 73–78, Bergen, Norway.

I. Dan Melamed. 2001. *Empirical Methods for Exploiting Parallel Texts*. The MIT Press.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Arjen Poutsma. 2000. Data-oriented translation. In *Proceedings of the 18th conference on Computational linguistics*, pages 635–641, Morristown, NJ, USA. Association for Computational Linguistics.

Markus Saers, Joakim Nivre, and Dekai Wu. 2009. Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *Proceedings of the 11th Internationa Conference on Parsing Technology (IWPT)*, pages 29–32.

Jörg Tiedemann and Gideon Kotzé. 2009. Building a large machine-aligned parallel treebank. In *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories (TLT'08)*, pages 197–208. EDUCatt, Milano/Italy.

Antal van den Bosch. 1997. *Learning to pronounce written words: A study in inductive language learning*. Ph.D. thesis, Maastricht University.

Vincent Vandeghinste and Scott Martens. 2009. Top-down transfer in example-based mt. In *Proceedins of the 3rd Workshop on EBMT*, pages 69–76.

Dekai Wu. 1995. Trainable coarse bilingual grammars for parallel text bracketing. In *3rd Annual Workshop on Very Large Corpora*, pages 69–82.

Ventsislav Zhechev and Andy Way. 2008. Automatic generation of parallel treebanks. In *Proceedings of the 22nd International Conference on Computational Linguistics (CoLing)*, pages 1105–1112.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141, Morristown, NJ, USA. Association for Computational Linguistics.